UbiqStor: Server and Proxy for Remote Storage of Mobile Devices

MinHwan Ok[1], Daegeun Kim[2], *Myong-soon Park[1]

[1]Dept. of Computer Science and Engineering / Korea University
Seoul, 136-701, Korea
[2]Network division, Mobile Communications / LG Electronics
Gyeonggi, 431-749, Korea
mhok@ieee.org, *myongsp@ilab.korea.ac.kr

Abstract. Mobile devices have difficulty in sustaining various services as in a wired environment, due to the storage shortage of the mobile device. The research[8] which provides remote storage service for mobile appliances using iSCSI has been conducted to overcome the storage shortage in mobile appliances. In research the proposed cache server performed well with relatively small files of sizes, however, did not perform well with large files such as database files, multimedia files, etc. The reason was the mobile device could not grasp the file as a whole and thus the cache server encountered frequent cache miss in spite of its huge buffer. In this paper we propose a proxy server that accommodates large files for mobile devices thus attains very high hit ratio.

1 Introduction

Mobile appliances are going to widen their region as years go. Many Efforts are performed to apply wire network environment services, which need large amount of storage space, such as multimedia and databases, to mobile appliances in wireless network environment. However, mobile appliances should be small and light to support mobility, so they use a small flash memory instead of a hard-disk of large data space. In the case of PDAs, these are usually equipped with RAM in size of 64~128MB. Cell-phones or smart phones permit smaller memory space. Therefore, there is difficulty preserving multimedia data such as mpg, mp3, etc. and installing large software such as database engines or using database. For mobile appliances limited storage space has been a barrier in applying various services of the wired environment. As a result, the shortage has driven development of remote storage services for mobile appliances, which can store large amounts of data, or provide various application services [5,8].

The bandwidth of wireless networks, which mobile devices use, is usually lower than those of wire networks. If the purpose of the remote storage service is to extend individual storage space for mobile appliances without data sharing, block-level I/O

* Corresponding Author

service is more suitable. iSCSI is a standard protocol that transports SCSI command. This is a representative block I/O through TCP/IP network. However, iSCSI has a problem that I/O performance drops sharply if network latency increases between iSCSI initiator and target. For the environment iSCSI-based remote storage service is applied to mobile appliances, we have achieved a breakthrough against the problem of iSCSI performance falling rapidly accordingly the mobile client recedes from the storage server and network latency increases[8].

The proposed cache server performed well with relatively small files, however, did not perform well with large files such as database files[1], multimedia files, etc. The reason was the mobile device could not grasp the file as a whole and thus the cache server encountered frequent cache miss in spite of its huge buffer. In this paper we propose a proxy server that accommodates large files for mobile devices thus attains very high hit ratio. In Sec. 2, we discuss the iSCSI basic operations to explain the reason why iSCSI performance drops down when distance between iSCSI initiator and target gets long. In Sec. 3, we propose ways to heighten hit ratio of iSCSI blocks through the proxy server, and present system architecture and a block management algorithm. We show and analyze simulation results with NS2 in Sec. 4. We conclude and describe the contribution in Sec. 5.

2 Background

2.1 iSCSI

The iSCSI (Internet Small Computer System Interface) is an emerging standard storage protocol that can transfer a SCSI command over IP network [5]. Since the iSCSI protocol can make clients access the SCSI I/O devices of a server host over IP Network, a client can use the storage of another host transparently without the need to pass through a server host's file system[6].

In iSCSI layer, which is on top of TCP layer, common SCSI commands and data are encapsulated in the form of iSCSI PDU (Protocol Data Unit). The iSCSI PDU is sent to the TCP layer for the IP network transport. Through this procedure, a client who wants to use a storage of the remote host, can use, because the encapsulation and the decapsulation of SCSI I/O commands over TCP/IP enable the client to access a storage device of the remote host directly [3]. Likewise, if we build a remote storage subsystem for mobile appliances based on iSCSI protocol, mobile clients can use the storage of a server host directly, like their own local storage. It enables mobile appliances to overcome the limitation of storage capacity, as well as the ability to adapt various application services of wired environment in need of mass scale data. Differently from the traditional remote storage subsystem, in file-level I/O service, iSCSI protocol provides block-level I/O. Thus it can make more efficient transmission throughput than the traditional remote storage subsystems, like CIFS and NFS.

Fig. 1 shows the exchange sequences of control and data packets in the read and write operation of iSCSI protocol [9, 10]. iSCSI exchanges the control packets (SCSI Command, Ready to Transmit, SCSI Response) and data packet (SCSI Data) to process one R/W operation.

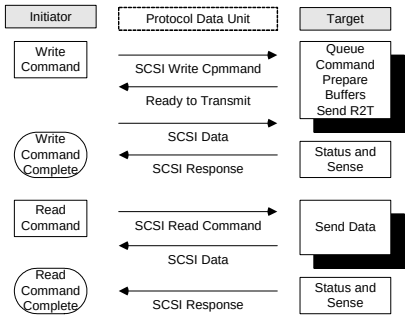


Fig. 1. iSCSI Basic Operation

Three control packets and one data packet are used in a write operation and spend 2 x RTT(Round-Trip Time). It takes 2 control packets and 1 x RTT for a read operation. Because the control packet, including header information, is no more than 48 bytes, bandwidth waste becomes big when the initiator recedes a little from the target. Therefore, the distance between iSCSI initiator and target, and the network latency influence iSCSI I/O performance.

2.2 Previous Works

Access from mobile devices to remote storage should be provided by a specific server. A storage server equipped with a vast capacity of storage devices has been developed[5] and it serves its storage volume to mobile devices on block-level I/O. Block service[2] provides necessary blocks of an opened file to the mobile device and updates corresponding blocks when modification occur. In developing the system we found that when iSCSI was applied to mobile appliances, iSCSI I/O performance fell rapidly if a iSCSI client had moved from the server to a far away location.

One way to alleviate this problem is to reduce the communication count between the mobile device and the storage server. iCache[4] is a research to improve iSCSI performance with local cache installed in the client system. Initiator's system has specific cache space for iSCSI data, and iSCSI block data are cached to minimize network block I/O. Therefore, iSCSI does not send I/O requests through the network every time the disk I/O happens. Instead it reads cached blocks or send blocks cached in LogDisk at once to the server for improving iSCSI performance. However it is difficult to apply iCache to mobile devices which lack memory, because iCache needs additional cache memory and hard-disk space to embody the local cache, Non-Volatile RAM and LogDisk, respectively.

The other way is to intervene between remote storage and mobile devices without adding internal storage into mobile devices. A cache server is introduced to shorten latency between the mobile device and the storage server[8], by prefetching next blocks and acknowledging block-recording early.

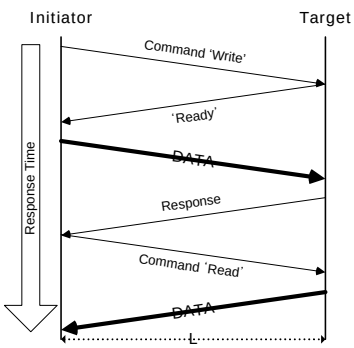


Fig. 2. Direct Communication between iSCSI Initiator and Target

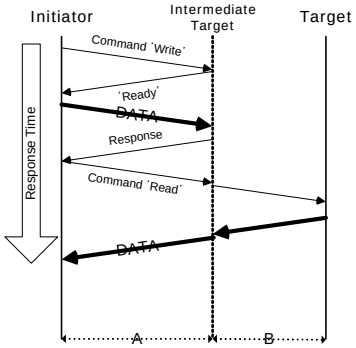


Fig. 3. Reduced Response Time by Intermediate Target

Fig. 2 and Fig. 3 show that response delay times are different when the iSCSI Initiator and Target communicate directly with each other, and when put with an iSCSI cache server. Three control packets should be exchanged to process a write command and two control packets for a read command. If the distance between initiator and target could be shorten, iSCSI I/O response time would be reduced. If the end nodes' processing delay of an iSCSI packet is ignored, response delay time is reduced by A/L when introducing an Intermediate Target as shown in Fig. 3.

The cache server is proposed to eliminate communication delay over wired links. Aggregating each next block and accessing on only block-level has a profit of small space requirement in the cache server, however it loses high hit ratio for mobile devices when an open file is so large that the application does not load the file as a whole. Database is a good example. In this paper we propose a relay system that is proxy for the storage server toward high hit ratio.

3 Ubiquitous Storage Proxy

In our previous work[8], we introduced a cache server which acts as an intermediate target for mobile devices. In this work, a strategy of the block management algorithm, caching next blocks, is amended to cache a whole file which bears requested blocks. We assume the following three factors to simplify the problem.

- Supposing a distributed storage server environment, iSCSI cache server spread over wide areas. A mobile client is connected with the nearest iSCSI cache through an iSNS(Internet Storage Name Server)[3].
- As transmission distance increases, propagation delay of physical media, and the sum of the queuing delay of intermediate routers increases.
- iSCSI latency includes propagation delay by distance, queuing delay and transmission delay by bandwidth, and ignores processing delay of end nodes.

3.1 iSCSI Proxy Server

In an iSCSI based remote storage service for mobile nodes, when a mobile node changes its service cell the distance between iSCSI initiator and target may be prolonged and packet transfer time can increase. In section 2, we showed that the SCSI command is processed sequentially in SAM-3 [7] and iSCSI basic operations need an exchange of several control packets to process one command. Because small control packets of the iSCSI protocol influence iSCSI response time, if the distance between the server and the client is long, the link utilization drops sharply and iSCSI performance becomes low. A proxy server intervenes between the storage server and a mobile device to reduce the packet transfer delay time and to heighten practical utilization of the network bandwidth. The nearest proxy server from mobile client is selected by iSNS and then the mobile device accesses remote storage via the proxy server, which also has an iSCSI connection with a remote storage server. The proxy server prefetches a whole file to be used by the mobile device for iSCSI read operations and gives immediate responses for iSCSI write operations. Therefore, response delay time of an I/O request shortens because the client has an iSCSI connection with a nearby proxy server, instead of a long-distant storage.

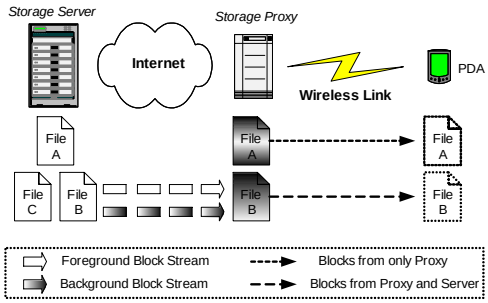


Fig. 4. The mobile device edits a file via the proxy server.

The strategy in caching is to prefetch the file as a whole that bears requested block. Fig. 4 illustrates read operations in the strategy. When a mobile device has requested a block opening a file A, the *Storage Proxy*, the iSCSI proxy server, relay the request to the *Storage Server*, where the files actually resides, and relay delivered requested block to *PDA*, the mobile device. We call this explicit block delivery as 'Foreground Block Stream'. As soon as the requested block delivered the Storage Server initiate loading the file of requested block into its local disk through its iSCSI buffer. The Storage Proxy has a size limit in initiating loading according to its capacity of storage devices attached. We call this implicit block delivery as 'Background Block Stream', and this stream uses idle resource of Storage Proxy's network interface. In the Fig. 4, file A is loaded as a whole thus the PDA requests any block in the loaded file, while some blocks of file B is delivered from the Storage Server as file B is still being loaded.

Fig. 5 illustrates write operations in the strategy. When a mobile device has requested recording a block, the *Storage Proxy* sends response to *PDA* then relay the request to the *Storage Server*. If the file should be flushed, closing a file A for exam-

ple, the block is added to ‘Background_List’ and the blocks in the list are delivered by Background Block Stream. The Storage Proxy periodically delivers blocks in the list and the Storage Server records the blocks delivered by Background Block Stream. If handover occurs, the mobile device wait until block recordings are completed by file closing to open the same file via another Storage Proxy. In this special case, the blocks are delivered by Foreground Block Stream.

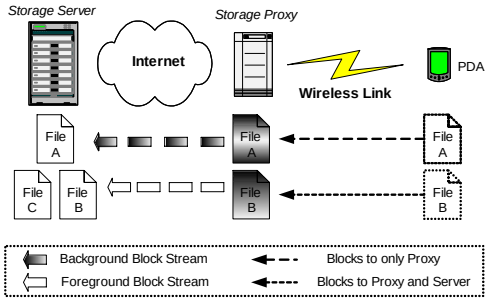


Fig. 5. The proxy server updates a file modified by the mobile device.

3.2 Proxy Architecture

Fig. 6 shows a module diagram of *UbiqStor service*. A proxy server consists of iSCSI initiator, target, and block management module. The proxy server has two iSCSI connections. One is a connection between the proxy server and a mobile device and the other is a connection with the storage server. The target module has an iSCSI session with a mobile device's iSCSI initiator and the initiator module has one with the iSCSI target module of the storage server. Two modules perform the same role, such as general iSCSI Target/Initiator module. However, the first iSCSI connection between mobile client and the proxy server is used for I/O requests of the client, and the latter is used in prefetching files from the storage server or in updating blocks at the storage server. The target/initiator module of the proxy server is controlled by block management module. iSCSI Buffer is managed as FIFO in the way to leave blocks used most recently.

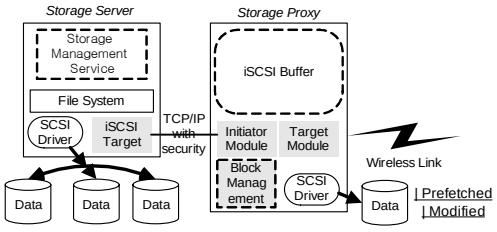


Fig. 6. UbiqStor Subsystem

Fig. 7 shows block management algorithm of the proxy server. In the case of a read operation, the block management module searches requested blocks in the iSCSI

buffer then the local disk to service the requested block. If the search fails, the initiator module of the proxy server sends the client's request to the storage server and receives the block to send to the client. In the case of write operation, the proxy server immediately sends a response message to the client and sends the blocks in appropriate time. Therefore the proxy server behaves nearly the storage server.

```
When Intermediate Target Receives an I/O Request
{
 If(Block Read)
 {//Client's I/O Request is Read Operation
    If( i exists in iSCSI Buffer )
    {
       Send i and Response to Initiator
    }
    Else
    {
       Send Read Request of i to Target
       Send i and Response to Initiator

      If( Background_Retrieving is not under
          progress )
      {   If( File_Size < THRESHOLD )
          { Initiate Background_Retrieving(BOF) }
      }
    }
 }
 Else
 {//Client's I/O Request is Write Operation
    Send Response to Initiator
    If( the Request is not 'Flush' )
    {
      Send Write Request of i to Target
    }
    Else
    {
     Add i at Tail of Background_List
    }
 }
}//End of When

Sub Background_Retrieving
{
    If( iSCSI Buffer is full or End-of-File )
    { Finish This Background_Retrieving }
    Else
    { Send Read Request of Next_pointer to Target }
    //Prefetch next block from Storage Server
}

Sub Background_Preserving
{
    If( Background_List is not empty )
    { Send Write Request of Next_Pointer to Target
      Remove Next_Pointer at Head of Background_List }
}
```

Fig. 7. Block Management Algorithm

According to the algorithm under the strategy, the hit ratio of requested blocks is 100% once the whole file is loaded into the proxy server. During file loading a requested block may be hit or not, and this is concerned with the size of a file being loaded. The number of files that are concurrently loaded is another factor since loading time gets longer as more files are being loaded together. We suppose the proxy server has two network interfaces, one for the storage server and the other for the mobile device. Therefore file loadings are not impacted by the communication between the proxy server and the mobile device.

4 Simulation

4.1 Simulation Configuration

Fig. 8 depicts the network of an iSCSI target of the storage server, intermediate target of the proxy server, and an initiator of the mobile device, which is simulated with Network Simulator 2.27 (NS2). The bandwidth of the wireless link between the mobile device and the proxy server is limited by 1Mbps (CDMA2000 1x EV-DO). The wired links between the proxy server and the storage server has a bandwidth of 2Mbps. Hit ratio of blocks is measured along the number of files concurrently being loaded. The distance from the mobile device to storage server, via the proxy server, is 200*km* and delay ratio of A and B is 7:3. Data size is 512 bytes which is SCSI block size used in PDAs under Windows CE, and which is one of the representatives in mobile appliances.

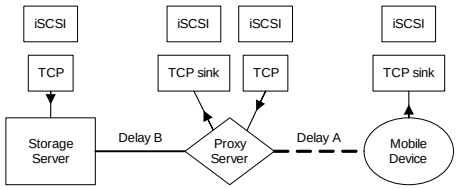
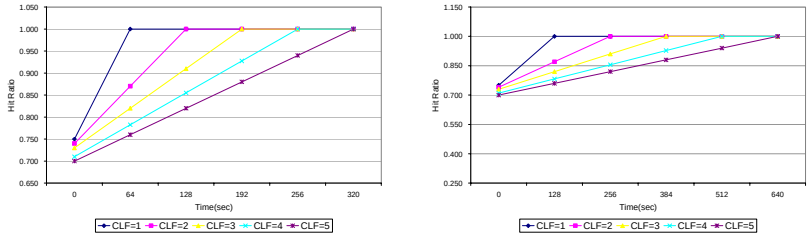


Fig. 8. Simulation Configuration within NS2

4.2 Hit Ratios by Data Types in Read operations

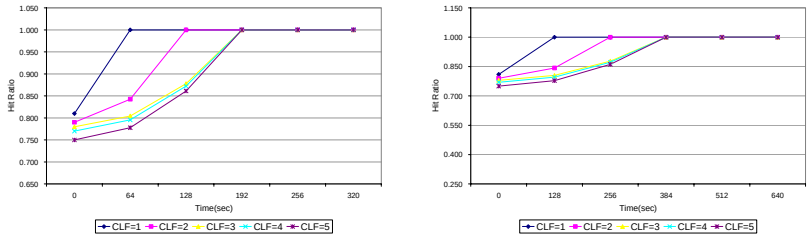
In the simulation with NS2, iSCSI throughput is influenced by a propagation delay, which depends on physical media, and transmission delay that depends on data size and bandwidth. In cases where iSCSI has a short delay, when the distance of iSCSI initiator from target is short, transmission delay is much longer than the difference of propagation delay by introducing the proxy server. Thus it has little influence on iSCSI performance. However, when the iSCSI initiator is distant from the target, the difference of propagation delay is much longer than transmission delay. Therefore the proxy server greatly improves the iSCSI throughput. Moreover the proxy server aims at the hit ratio of 100% by accommodating a whole file. However it takes long time to

load large files in sizes of 16MB, 32MB, and so on. The simulation focuses on the hit ratio of blocks in reading from those files until they are completely loaded.



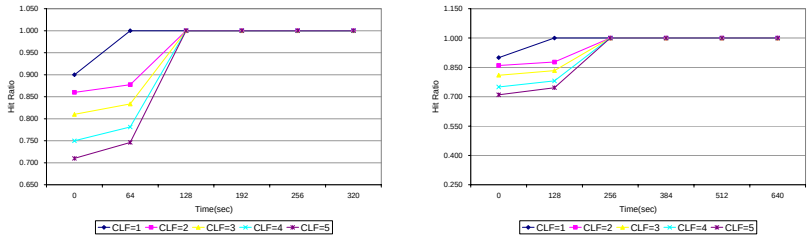
(a) Application File of 16Mbytes

(d) Application File of 32Mbytes



(b) Document File of 16Mbytes

(e) Document File of 32Mbytes



(c) Multimedia File of 16Mbytes

(f) Multimedia File of 32Mbytes

Fig. 9. Hit Ratios in Loading Files

We used three file types, Application file, Document file, and Multimedia file. Application file is constituted by randomly accessing requests, i.e., database file. Multimedia file is constituted by sequentially accessing requests, i.e., MPEG file. Document file is constituted by mixes of random and sequential accessing requests. Files of 16MB are being loaded as shown in Fig. 9. When the proxy server is loading only one file, it takes 64*sec* to the 100% hit ratio for one Application file. Since blocks are requested randomly, hit ratio has reached 100% at the time the file is completely loaded. File loadings take longer time proportional to the number of files concurrently loaded, CLF in the figure. For Document file, however, the same CLFs take less time to reach the hit ratio of 100%. File loadings take the same time as in the case of Application File. Document files have partially sequential access patterns and they has not requested a file as a whole when editing from the mobile device. Timely necessary blocks are requested at random, but as a series of blocks. Although file loadings take the same time as in the case of Application File, Multimedia files reache the hit ratio of 100% even earlier than those in the case of Document File. The reason is their sequential access patterns. For each case of file types, it takes about 64 seconds in loading only one file and 1MB of a file is loaded in just 4 seconds. That loaded front

part of 1MB suffices ordinary use tendency since a user usually does not request every block of 16MB in several seconds, in all cases including the case of Application File. This is why the hit ratios are high from the starts. Larger files take proportionally longer loading times, and hit ratios follow similar aspect by the data type, in loading files of 32MB.

5 Conclusion

In our previous work[8], we described efficient ways introducing iSCSI cache server, to achieve a breakthrough against the problem of iSCSI performance falling when applying iSCSI-based remote storage services for mobile appliances. In this work we developed the cache server into a proxy server to improve iSCSI buffer management to heighten block hit rate. Throughout the simulations the hit ratio has greatly grown in the cases of large files. Although not presented in this paper, file loadings need not take longer time proportional to its size. The proxy server has dozens of 2MB bandwidth the on wire network, while the mobile device has only limited bandwidth up to 1Mbps over the wireless network. Thus file loading can be completed much earlier on the wire network, and hit ratio can reach 100% earlier.

Preserving files in some server and retrieving them out of any server is necessary requirement for ubiquitous computing. We developed remote storage server and proxy server that supply ubiquitous storage, UbiqStor. The system has shown its effectiveness by simulation and much of mobile appliances should be forwarded to ubiquitous computing by UbiqStor service.

Reference

1. Shepard SJ. Embedded Databases Can Power Emerging World of Information to Go. *IEEE Distributed Systems Online*.
2. Block Device Driver Architecture, http://msdn.microsoft.com/library/en-us/wceddk40/html/_wceddk_system_architecture_for_block_devices.asp
3. Clark T. *IP SANs: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks.* Addison-Wesley, 2002
4. He X, Yang Q, Zhang M. A Caching Strategy to Improve iSCSI Performance. *Local Computer Networks* 2002: 278-285.
5. Park S, Moon B, Park M. Design, Implementation, and Performance Analysis of the Remote Storage System in Mobile Environment. *Information Technology & Applications* 2004.
6. Lu Y, Du DHC. Performance Study of iSCSI-Based Storage Subsystems. *IEEE Communication Magazine*; 41(8): 76-82.
7. SAM-3 : Information Technology - SCSI Architecture Model 3. Working Draft. T10 Project 1561-D. Revision 7, 2003.
8. Ok MH, Kim D, Park M. UbiqStor: A Remote Storage Service for Mobile Devices. *Parallel and Distributed Computing: Applications and Technologies* 2004: 685-688.
9. Meth KZ, Satran J. Design of the iSCSI Protocol. *Mass Storage Systems and Technologies* 2003, 116-122.
10. Satran J. iSCSI Draft20, http://www.ietf.org/internet-draft/draft-ietf-ips-iscsi-20.txt